VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozpoznání obličejů a jejich sledování na procesorech i.MX

Face Detection and Tracking on the i.MX Processors

2018 Jan Kněžík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Kněžík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Rozpoznání obličejů a jejich sledování na procesorech i.MX
Face Detection and Tracking on the i.MX Processors

Jazyk vypracování: čeština

Zásady pro vypracování:

Pod označením i.MX se skrývá rodina aplikačních procesorů, které jsou založeny na architektuře ARM. Kombinace vysokého výkonu těchto procesorů s možností dalšího rozšíření nabízí ideální řešení například pro automobilové a průmyslové využití. Cílem této práce bude vytvoření aplikace pro rozpoznání obličejů právě na procesorech i.MX.

1. Seznamte se se základními pojmy v oblasti detekce objektů v obrazech, zejména se zaměřte na detekci a rozpoznání lidských tváří.
2. Seznamte se s procesory i.MX.
3. Prozkoumejte jaké možnosti nabízí knihovna OpenCV v oblasti detekce, rozpoznání a sledování lidských tváří.
4. Vytvořte aplikaci (s využití knihovny OpenCV), která poběží na procesorech i.MX a bude detekovat a rozpoznávat lidské tváře v obrazech za pomocí kamery.
5. Experimentálně ověřte přesnost a rychlost řešení.
6. Své závěry řádně zdokumentujte v textu práce.

Seznam doporučené odborné literatury:

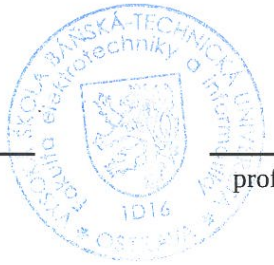
[1]Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. vol. 1, pp. I-511-I-518 vol.1 (2001)
[2]Zhao, W., Chellappa, R., Phillips, P., and Rosenfeld, A.: Face recognition: A literature survey. ACM Computing Surveys (CSUR), pp. 399-458 (2003)
[3]Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S.Z.: Learning multi-scale block local binary patterns for face recognition. In: ICB. pp. 828-837 (2007)

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radovan Fusek, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 30.04.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 26. dubna 2018 .....

Rád bych poděkoval vedoucímu bakalářské práce Ing. Radovanu Fuskovi, Ph.D. za odborné vedení a cenné rady při zpracování této práce.

Abstrakt

Tato bakalářská práce se zabývá detekcí a rozpoznáním lidských tváří. Nejdříve je provedena rešerše známých a efektivních metod. Patří zde starší průlomové metody jako Viola-Jones u detekce nebo Eigenfaces u rozpoznání, ale také moderní metody jako neuronové sítě. Posléze jsou metody implementovány s pomocí OpenCV knihovny a otestovány pro nalezení optimálních parametrů na databázích FDDB u detekce obličejů a AT&T u rozpoznání obličejů. S těmito parametry jsou metody otestovány pomocí kamerového modulu Raspberry Pi v2 v reálném prostředí. Všechno testování proběhlo na zařízení Raspberry Pi 3 model B s procesorem i.MX.

Klíčová slova: počítačové vidění, strojové učení, detekce obličejů, rozpoznání obličejů, extrakce příznaků, klasifikace, Viola-Jones, Eigenfaces, Fisherfaces, Lokální binární vzory, OpenCV

Abstract

This bachelor thesis deals with detecting and recognizing human faces. First of all, known and effective methods are described, which include older breakthrough methods like Viola-Jones for detection or Eigenfaces for recognition, but also modern methods like neural networks. After that, methods are implemented using OpenCV libraries and tested to find optimal parameters on FDDB database for face detection and on AT&T database for face recognition. With these parameters, the methods are tested using the Raspberry Pi v2 camera module in a real environment. All testing was carried out on device Raspberry Pi 3 model B with i.MX processor.

Key Words: computer vision, machine learning, face detection, face recognition, feature extraction, classification, Viola-Jones, Eigenfaces, Fisherfaces, Local binary patterns, OpenCV

Obsah

Seznam použitých zkratek a symbolů

SVM	–	Support vector machine
HoG	–	Histogram of oriented gradients
k-NN	–	k-nearest neighbors
LBP	–	Local binary pattern
LBPH	–	Local binary pattern histogram
PCA	–	Principal component analysis
LDA	–	Linear discriminant analysis
CNN	–	Convolutional neural networks
FDDB	–	Face detection data set and benchmark
XML	–	Extensible markup language
API	–	Application programming interface

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Počítačové vidění je odvětví výpočetní techniky, u kterého je úkolem získat informace o reálném světě pomocí digitálního obrazu. To zahrnuje také detekci a rozpoznávání obličejů. Tyto úkoly jsou pro člověka jednoduché, ale přijít s metodou jak popsat obličej počítači, už tak jednoduché není. Tato oblast výpočetní techniky se v poslední době čím dál více rozšiřuje, díky růstu výpočetního výkonu a poptávky po systémech zahrnující právě detekci či identifikaci. Detekce je většinou prvním krokem pro rozpoznání tváře, ale je také užitečná i sama o sobě. V dnešní době lze najít detekci v každém fotoaparátu pro zaostření tváře. Rozpoznání člověka pomocí obličeje je jedna z mnoha biometrických metod. První spolehlivou metodou, jak identifikovat jedince, byl otisk prstu. Avšak jedna z mnoha výhod u rozpoznání pomocí obličeje je, že není potřeba participace daného jedince. Existuje také mnoho nevýhod a i v dnešní době nejsou systémy bezchybné. V průběhu času se lidská tvář mění a i drobné změny, jako jsou brýle, vousy, natočení hlavy nebo špatné osvětlení, drasticky zhorší výsledek. Velká poptávka rozpoznávání tváří je v sektoru bezpečnosti, například na letištích nebo obchodech. V dnešní době obličej také nahrazuje verifikaci uživatele, místo hesla.

Cílem práce je popsání efektivních a široce používaných metod, které vstoupili do historie jako důležité milníky a které sloužili jako stavební kámen pro rozvoj detekce a rozpoznání obličejů, ale také metody používané v dnešní době. Dále jsou metody implementovány a otestovány s pomocí knihovny OpenCV, na zařízení s i. MX procesorem.

V druhé kapitole se seznámíme s metodami pro detekci obličejů. Metody jsou rozdělené podle přístupu a u každé kategorie je uveden příklad. Nejdůležitější je tu však metoda Viola-Jones.

V třetí kapitole je na úvod vysvětlený obecný proces rozpoznání obličeje a následně jsou rozebrány jednotlivé metody. Neuronové sítě jsou popsány do větší hloubky, jelikož jsou v současnosti nejvíce používané.

Ve čtvrté kapitole se seznámíme s použitým hardwarem. Nejdříve jsou obecně popsány i.MX procesory a následně použité zařízení Raspberry Pi a Raspberry Pi kamerový modul.

V páté kapitole je popsána OpenCV knihovna a její použité moduly pro implementaci metod. Také jsou vysvětleny nejdůležitější části u implementace a hlavně parametry ovlivňující výkon metod.

V šesté kapitole je vysvětlený způsob hodnocení implementovaných metod a následně jsou zobrazeny výsledky.

2 Detekce obličejů

Metody detekce obličejů nejsou pevně rozdělené do kategorií, ale lze je rozdělit podle metod založených na znalostech o lidské tváři, neměnných rysech, porovnání šablon a vzhledově založené metody [1]. Některé metody však přesahují hranice mezi kategoriemi.

2.1 Metody založené na znalostech o lidské tváři

Tyto metody zachycují znalosti o lidské tváři a popisují je jako seznam pravidel. Pravidla jsou založena na pozici rysů obličeje a jejich vztahů. Například obličej má dvě oči, které jsou si symetrické a oblast očí je tmavší než spodní oblast pod očima. Vymyslet dobře definovaná pravidla je ale obtížné. Pokud jsou pravidla příliš přísná, ne všechny obličeje budou detekovány a pokud by byla příliš obecná, došlo by k falešným detekcím.

Yang a Huang použili hierarchickou metodu založenou na znalostech, která se skládá ze tří úrovní pravidel [1]. Na nejvyšší úrovni jsou na základě obecných pravidel navrženi všichni možní kandidáti. Obraz je tedy převedený do stupňů šedi a nižšího rozlišení a poté je porovnán s maskou, pomocí které se rozhodne jestli má obraz správné hodnoty jasu v určitých oblastech. Tito kandidáti se posílají do druhé úrovně, kde se provede ekvalizace lokálních histogramů a následně detekce hran. V poslední úrovni jsou na obrysech obličeje z druhé úrovně prováděna striktní pravidla, která rozhodnou jestli se v obraze nachází např. nos, oči a pusa.

2.2 Metody založené na neměnných rysech obličeje

Cílem metod v této kategorii je najít rysy obličeje, které se nemění v nežádoucích světelných podmínkách a různých pózách. Takové rysy jsou například textura a barva kůže.

I když mají lidé různé barvy pleti, je to i tak efektivní neměnný rys obličeje. U detekce na základě barvy kůže jsou nejprve určené rozsahy v barevném modelu, které odpovídají barvě kůže. Podle metody je vybraný barevný model RGB, HSV, nebo také YCrCb. Po segmentaci obrazu na základě rozsahu barvy je výsledkem obraz, kde by se měla vyskytovat pouze kůže. To ale neznamená, že všechna kůže je pleť obličeje. Proto se dále aplikuje například detekce hran pro odstranění pozadí. Tato metoda většinou slouží jako pomocná metoda pro přiblížení, kde by se na obrazu mohl vyskytovat obličej.

2.3 Metody založené na porovnání šablon

Šablony těchto metod předepisují vzor obličeje jako funkci a mohou být použité na celý obličej, nebo na jednotlivé rysy obličeje. Detekce je realizovaná na základě vypočtu korelace mezi vstupním obrazem a šablonami. Metody jsou jednoduché na implementaci, ale zpravidla jsou neefektivní při změně pózy, tvaru a měřítka.

Sinha navrhl metodu, která používá invariantní šablony pro popis vzoru obličeje [1]. Základní myšlenka je, že i když se změní osvětlení v obrazu, tak relativní osvětlení určitých oblastí obličeje

zůstanou stejné. Šablona se skládá z oblastí a jejich vzájemných vztahů, které reprezentují poměr jasů mezi dvěma oblastmi. Šablona detekuje obličej, pokud dostatek poměrů přesáhne nastavený práh.

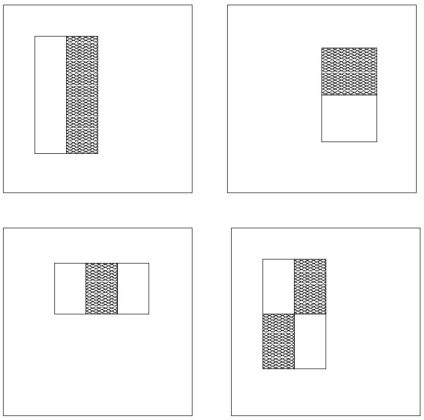
2.4 Metody založené na vzhledu

Metody založené na vzhledu vychází ze strojového učení a statistické vědy. Oproti metodám založené na porovnání šablon, které jsou předefinované ručně, jsou tyto metody trénované z trénovacích dat. Trénovací data obsahují mnoho obrazů s obličeji a ne-obličeji.

2.4.1 Viola-Jones metoda

Viola-Jones detektor, navržen v roce 2001 [2], je významná metoda, která byla jako první robustní a dostatečně rychlá pro detekci v reálném čase. Hlavní myšlenkou je popsat trénovací obrazy s Haar příznaky, vybrat nejdůležitější příznaky a vytvořit z nich klasifikátory pomoci algoritmu AdaBoost a nakonec sestrojit kaskády z daných klasifikátorů.

Haar příznaky
Výpočet příznaků a jejich název vychází z Haar vlnek. Existuje mnoho druhů příznaků, ale pro detekci obličeje jsou v původní práci použity 4 příznaky; dva dvou-obdélníkové, jeden tří-obdélníkový a jeden čtyř-obdélníkový (obrázek 1). Výpočet příznaku se provede součtem hodnot jasů pixelů v bílé a černé oblasti a tyto sumy se od sebe odečtou. Příznak tedy reprezentuje jedna hodnota.
Pro detekci je ve vstupním obrazu vytvořeno detekční okno, které se posouvá po celém obraze. Po každém cyklu obraz změní svojí velikost, aby mohlo okno detekovat obličeje různých velikostí. Svoji velikost může také měnit místo obrazu samotné detekční okno. Příznaky se v detekčním okně posouvají horizontálně a vertikálně a po každém cyklu zvětšují svojí velikost. U detekčního okna 24x24 px existuje pro 4 Haarovi příznaky více než 160 000 kombinací [3]. Kvůli obrovskému počtu kombinací příznaků zjednodušíme jejich vypočítání pomocí integrálních obrazů.



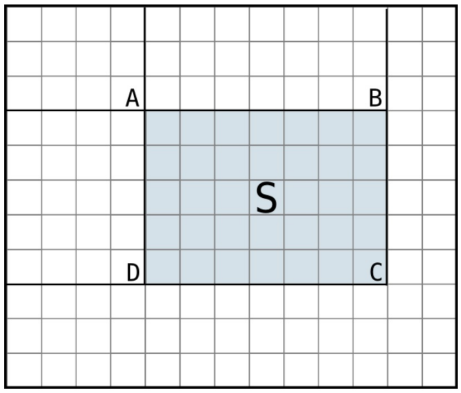
Obrázek 1: Haar příznaky [3]

Integrální obraz

Obraz se převede na integrální obraz, ve kterém jde suma každého obdélníku v obraze vypočítat nanejvýš pomocí 3 operací. V integrálním obraze je každý pixel součet sebe samého a všech pixelů od něho nalevo a nahoru:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

kde ii(x,y) je integrální obraz a i(x',y') je originální obraz [3].
Nyní jde suma obdélníku v obraze jednoduše vypočítat pomocí S=A+C-B-D, jak je vidět na obrázku 2. Ale i s tímto vylepšením je metoda příliš pomalá, jelikož obsahuje hodně Haar příznaků. Proto je použit algoritmus AdaBoost.

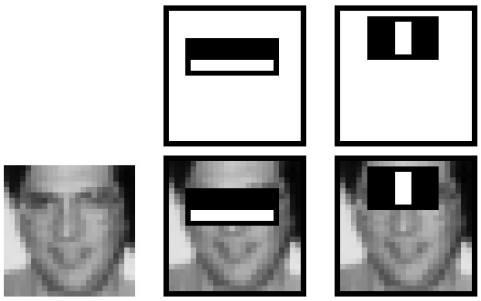


Obrázek 2: Znázornění hodnot v integrálním obraze pro výpočet oblasti S [15]

AdaBoost

Adaptive boosting je algoritmus, který vybere Haar příznaky, které nesou důležité informace pro detekci a vytvoří tak slabé klasifikátory v době tréninku. Tyto binární klasifikátory mají výstup

1, pokud si myslí, že detekovaly obličej a 0 pokud nikoliv. Jsou vytvořeny kombinací příznaku, prahu a parity.
U každého cyklu algoritmu je upravována váha obrazů, podle chybovosti klasifikátorů na daném obraze. Čím větší chybovost na obraze, tím má větší váhu a vliv na klasifikátory. Následně je upravován práh klasifikátoru, aby minimalizoval svoji chybovost. Klasifikátory s nejmenší chybovostí jsou vybrány jako slabé klasifikátory. Na obrázku 3 jsou vidět první dva vybrané příznaky využívající toho, že nos je světlejší než okolní tváře a že okolí očí je tmavší než oblast pod očima [3]. Vybrané klasifikátory jsou nazvané slabými, jelikož sami nedokážou klasifikovat, jestli se jedná o obličej. Naopak tyto klasifikátory jsou pouze o něco lepší, než náhodné hádaní. Proto AdaBoost lineárně kombinuje slabé klasifikátory do silného klasifikátoru a ten je schopný detekovat obličeje. Je možno vybrat různě velkou množinu příznaků, ale pro optimální detekci použili Viola a Jones ve své práci [3] těch 2500 nejlepších.



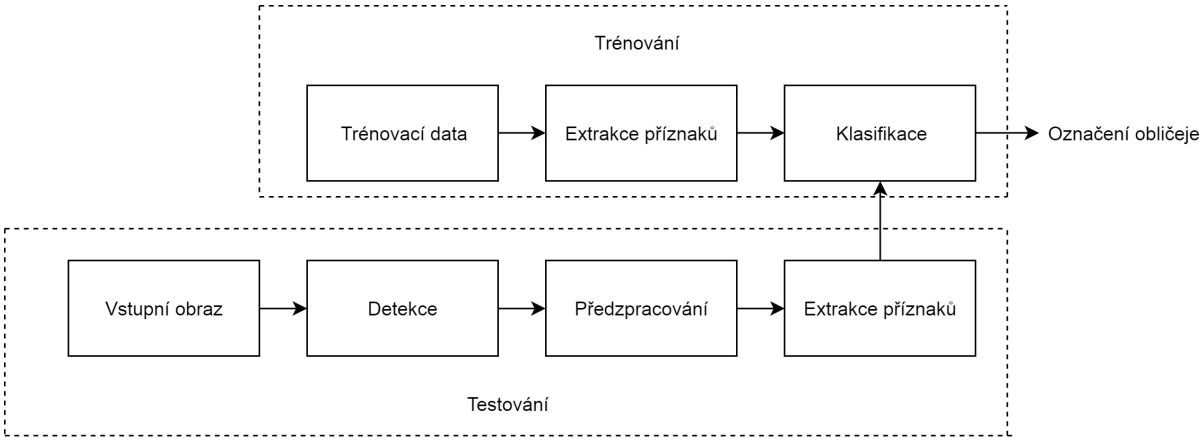
Obrázek 3: První dva vybrané příznaky algoritmem AdaBoost a jejich použití [3]

Kaskádování klasifikátorů

Většina detekčních oken je vyhodnocena jako negativní a nemusí vůbec nést podobné vlastnosti obličeje. Pro zavrhnutí takových oken není nutné použití všech klasifikátorů.
Klasifikátory jsou rozdělené do kaskád, kde první kaskáda může obsahovat i pouze 2 klasifikátory, které zahodí většinu pozadí. Počet klasifikátoru v následujících kaskádách narůstá a tak i jejich schopnost přesnější klasifikace. Každá kaskáda tedy zavrhuje okno, nebo posílá dále možný obličej. Pokud projde okno všemi kaskádami, je prohlášeno za detekovaný obličej.

3 Rozpoznání obličejů

Nejdříve se seznámíme s procesem identifikování obličeje a pár klíčovými slovy. Proces rozpoznání obličeje lze obecně rozdělit do několika kroků (obrázek 4). Předtím než můžeme začít se samotným rozpoznávaním obličejů, je nutno provést trénovací fázi. V trénovací fázi proběhne extrakce příznaků na všech trénovacích obrazech a na základě těchto příznaků bude klasifikátor indentifikovat obličeje testovacích obrazů. Po natrénování modelu se může provádět testovací fáze, kde se musí na vstupním obrazu najít oblast, kde se nachází obličej, jak bylo popsáno v minulé kapitole. Oříznutý obraz může být dále předzpracován, například zarovnáním podle očí nebo převodem do stupňů šedi. Poté se extrahují příznaky, stejně jako při trénování a podle těchto příznaků bude obraz klasifikován. Mnoho metod pro rozpoznání obličejů lze také použít u detekce obličejů.



Obrázek 4: Trénovací a testovací fáze zobrazena v pořadí kroků

Extrakce příznaků

Z obrazu se musí extrahovat důležité informace o obličeji, resp. vlastnosti obličeje, které se ve zpracovaní obrazů nazývají příznaky. Vlastnost obličeje si můžeme představit jako dlouhý nos nebo malé rty. Počítač si ovšem vlastnosti obličeje interpretuje vlastním způsobem, podle dané metody, aby jim rozuměl on. Výsledek extrakce je vektor příznaků, který reprezentuje daný obličej. Tento vektor se nachází v prostoru příznaků.

Extrakce příznaku je také spojena s redukci dimenzí, která redukuje redundantní informace o obličeji. To je kromě snížení výpočetní náročnosti velice důležitá vlastnost, kvůli prokletí dimenzionality. Tento fenomén vysvětluje, jak s narůstajícími dimenzemi exponenciálně roste velikost prostoru příznaku a po určitém bodě začnou extra příznaky pouze škodit. To se stane, jelikož prostor příznaků začne být řídký. Na jednu stranu se dají v řídkém prostoru jednodušeji klasifikovat data, ale nastane přetrénování modelu. U přetrénování je klasifikátor naučený příliš specificky na trénovacích datech a selže na dosud nespatřených obrazech. Navíc u klasifikátorů,

které používají euklidovskou vzdálenost, vypadají vektory v prostoru s velkým počtem dimenzí skoro stejně. Aby se tomuto předešlo, musí být použita již zmíněná redukce dimenzí nebo musí exponenciálně růst počet trénovacích obrazů, v poměru s počtem příznaků.

Klasifikace
Ve statistice se kategoriím, nebo v našem případě osobám, říkají třídy. Úkolem klasifikátoru je rozhodnout, které třídě patří vektor příznaků vstupního obrazu. U rozpoznání obličeje se jedná o multitřídní klasifikaci s předpokladem, že se v databázi nachází více než 2 osoby. Výkon klasifikátoru mimo jiné záleží na balanci počtu vektorů na třídu a počtu příznaků.
Učení klasifikátoru se rozděluje na učení s učitelem a bez učitele. Pokud mají trénovací data určený požadovaný výstup, do které třídy vektor patří, jedná se o učení s učitelem. Čím jsou vektory příznaků stejné třídy více seskupené dohromady v prostoru příznaků, tím je jednodušší a přesnější klasifikace. Učení bez učitele nemá označené trénovací data, takže jsou vektory příznaků seskupené na základě vzájemné podobnosti. Výsledek algoritmu není až tak klasifikace, jako shlukování dat.

3.1 Eigenfaces

Před Eigenfaces se metody pro rozpoznání obličejů zaměřovaly na jednotlivé rysy obličeje jako nos, ústa, oči a definovaly model pomoci pozic, velikostí a vztahů mezi těmito rysy. Eigenfaces zastává holistický matematický přístup a snaží se najít, které informace v obraze jsou důležité. Toho dosahuje pomocí metody Principal component analysis. Tato metoda redukuje dimenze trénovacích dat a zanechá pouze důležité informace pro rozpoznání obličejů.

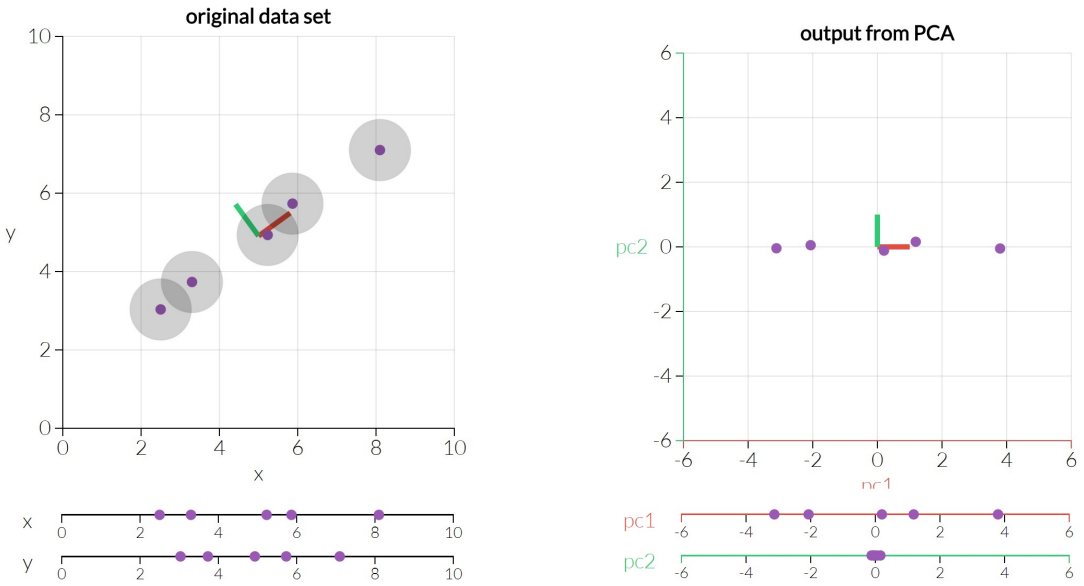
Z kovariační matice trénovacích dat se vypočítají vlastní vektory a jejich vlastní hodnoty [4]. Vlastní vektory resp. hlavní komponenty, jsou osy nového prostoru příznaků, které zachycují variaci mezi trénovacími obrazy. Čím větší vlastní hodnotu má vlastní vektor, tím větší nese variaci dat. Maximální počet vlastních vektorů je počet trénovacích obrazů. Konečný vektor příznaků je tedy potenciálně obrovský a nevhodný pro klasifikaci, proto se redukují dimenze vybráním pouze určitého počtu vlastních vektorů s největšími vlastními hodnotami.
Každý vlastní vektor má stejný počet dimenzí jako trénovací obrazy, protože každá část obrazu více či méně přispívá ke každému vektoru. Z toho důvodu vlastní vektory mohou připomínat obličeje a proto se nazývají eigenfaces. Na obrázku 5 jsou vidět první 4 eigenfaces s největší vlastní hodnotou z AT&T databáze.

Obrázek 5: Znázornění 4 eigenfaces s největší vlastní hodnotou

Každý obličej je reprezentovaný lineární kombinací eigenfaces. Konečný vektor příznaků pro každý obraz je tedy vektor vah, kde každá váha náleží jednomu eigenface. Více informací ohledně výpočtu lze najít v práci "*Eigenfaces for Recognition*"[4].

Příklad

Na obrázku 6a jsou vidět trénovací data o dvou dimenzích. Tento prostor příznaků zmenšíme na jednu dimenzi a zároveň si necháme nejdůležitější informace o trénovacích datech (obrázek 6b). V tom případě se hlavní komponentou s největší vlastní hodnotou se stane *pc1* a zároveň se stane novou osou x, nového prostoru příznaků. Hlavní komponentu *pc2* zahazujeme, jelikož má menší vlastní hodnotu a v našem případě nenese skoro žádné informace.



(a) Originální trénovací data

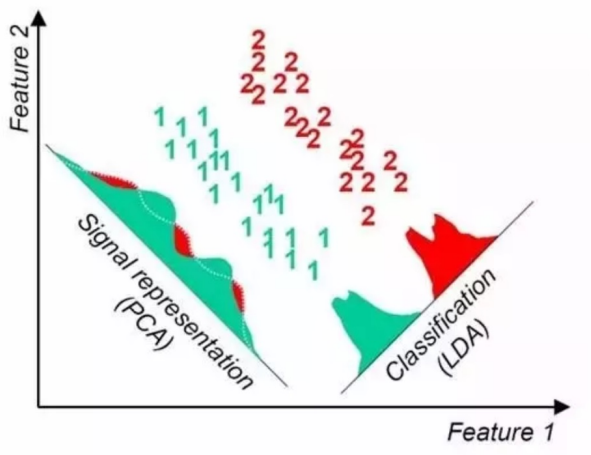
(b) Výstup PCA

Obrázek 6: Znázornění PCA [16]

3.2 Fisherfaces

Fisherfaces se snaží s metodou Linear discriminant analysis, najít lineární projekci trénovacích dat do prostoru příznaku s malým počtem dimenzí, podobně jako Eigenfaces. Oproti Eigenfaces,

ale mají obecně lepší výkon při změně směru světla a výrazů tváře [5], protože se metoda nesnaží jenom co nejefektivněji zachytit důležité informace, ale také vytvořit rozptyl dat více spolehlivých pro klasifikaci. Toho docílí projekci do prostoru, kde je rozptyl mezi třídami maximální, ale zároveň je rozptyl v jednotlivých třídách minimální. Eigenfaces hledají pouze projekci, která maximalizuje variaci dat mezi všemi obrazy. Takže u Fisherfaces změna kvůli osvětlení, nebo výrazu tváře u obrazů stejných osob, nebude větší než změna kvůli struktury obličeje u obrazů rozdílných osob. Na druhou stranu je výkon Eigenfaces v porovnání s Fisherfaces lepší, je-li v trénovacích datech malý počet obrazů na osobu. V ukázce na obrázku 7 je vidět, že i když PCA zachytilo největší variaci všech dat, třídy se staly lineárně neoddělitelné. Oproti tomu LDA zachytilo nejmenší změnu dat ve třídě a třídy jsou jednoduše lineárně oddělitelné. Více informací ohledně výpočtu lze najít v práci "*Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*"[5].

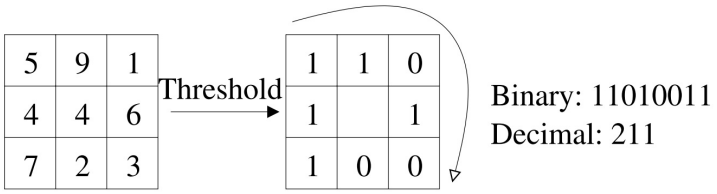


Obrázek 7: Rozdíl zachycení dat mezi PCA a LDA [17]

3.3 Lokální binární vzory

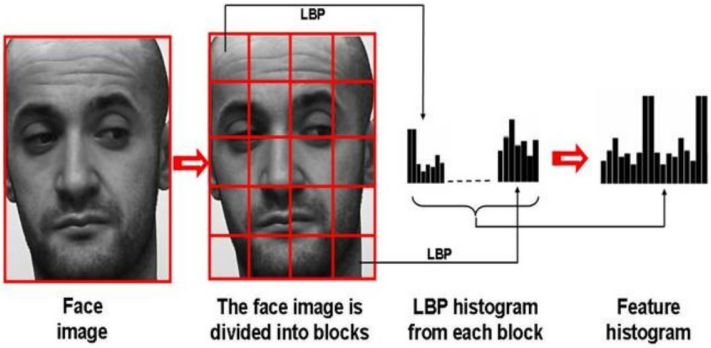
Lokální binární vzory (Local binary patterns) jsou metoda, originálně použita pro popis textury. V roce 2006 byl LBP operátor navržen pro popis obličeje [6]. Pro redukci dimenzí a popis obličeje se zde nepoužívají holistické metody, ale místo toho se každý, vyjímaje krajních pixelů, zpracuje do binárního kódu. Četnost všech těchto kódu resp. globální histogram reprezentuje vektor příznaků, který obsahuje informace o rozdělení lokálních vzorů.

Nejprve se obraz převede do stupně šedi. Pro vypočítání binárního kódu pixelu se vytvoří 3x3 okno, ve kterém se hodnota prostředního pixelu porovná s jeho 8 sousedy. Pokud je hodnota prostředního pixelu vetší než souseda, výsledek porovnání je 0 a pokud je menší, tak je výsledek 1. Výsledky dohromady vytvoří 8-bitový binární kód, který je přiřazený prostřednímu pixelu (obrázek 8). Tento proces se opakuje pro všechny, kromě krajních pixelů, jelikož nemají všechny sousedy.



Obrázek 8: Znázornění výpočtu binárního kódu [6]

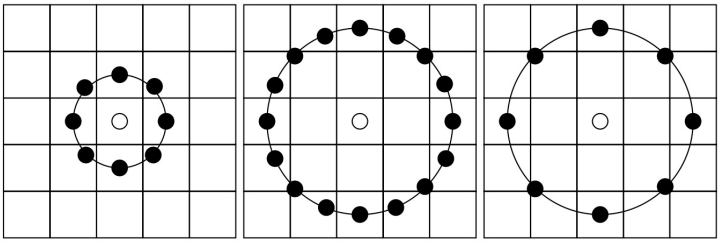
U popisu obličeje se obraz navíc rozdělí do bloků, pro zachování prostorových informací. Zmenšením velikosti bloku narostou výpočetní náklady a konečný vektor příznaků, ale při moc velkém bloku dojde ke ztrátě prostorových informací. Z nových hodnot pixelů se v každém bloku vytvoří histogram. Nakonec se tyto histogramy spojí dohromady do globálního histogramu, který popisuje celý obličej (obrázek 9).



Obrázek 9: Proces LBPH [18]

Kruhové LBP

Později byl operátor rozšířen o použití symbolů R a P, značený $LBP_{R,P}$. Hodnota R odpovídá vzdálenosti sousedů od prostředního pixelu a P je počet sousedů v daném okolí (obrázek 10). Použitím kruhového rozmístění sousedů a bilineární interpolaci hodnot pro souřadnice pixelů s necelými čísly, může být použit libovolný poloměr a počet sousedů.

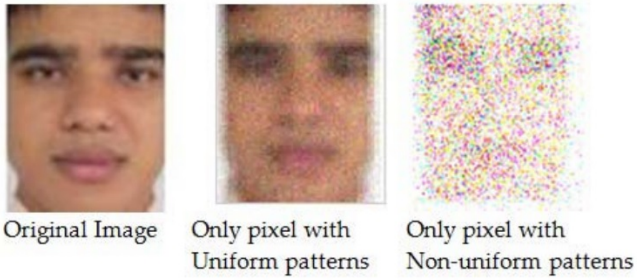


Obrázek 10: Kruhové sousedství $LBP_{8,1}$, $LBP_{16,2}$ a $LBP_{8,2}$ [7]

Uniformní vzor

Histogram 8-bitových binárních kódu má $2^8 = 256$ pozic. Tento počet může být ale zmen-

šený, použitím uniformních vzorů. Uniformní vzor obsahuje nanejvýš dva přechody z 0 na 1 nebo z 1 na 0. Zbytek vzorů se označí jako jeden. Jelikož 8-bitový binární uniformní vzor má maximálně 58 kombinací, výsledný histogram bude mít pouze 59 pozic. Tohle umožní velkou paměťovou úsporu, na úkor malé ztráty informací o obličeji. Na obrázku 11 je vidět rozdíl ztráty informací mezi uniformními a ne-uniformními vzory. Ahonen a kolektiv zjistili, že u jejich $LBP_{16,2}$ bylo ze všech vzorů 79,3% uniformních [6].



Obrázek 11: Rozdíl mezi uniformními a ne-uniformními vzory [8]

3.4 Histogram orientovaných gradientů

Histogram orientovaných gradientů (Histogram of oriented gradients) je deskriptor příznaků, který počítá četnost orientace gradientů v lokálních částech obrazu. Základní myšlenkou je popsání objektu podle těchto gradientů, které reprezentují změny směrů hran. Hrany se detekují změnou intenzity sousedních pixelů. U metody není potřeba normalizace barev obrazu, protože je již součástí deskriptoru.

Nejprve se vypočítají gradienty u každého pixelu. Jednoduchou a také efektivní metodou je použití derivační masky [-1,0,1], aplikovaná zvlášť pro horizontální a vertikální směr [9]. Pokud se pracuje s barevnými obrazy, tak je gradient vypočítán pro všechny tři kanály RGB a v potaz je brán pouze gradient s největší velikostí. Obraz je rozdělen do buněk a pro každou buňku je vytvořen histogram. Kanály histogramu budou reprezentovat rovnoměrně rozdělenou orientaci gradientu, od 0–180° pro gradienty bez znaménka a 0–360° se znaménkem. V tabulce 1 je vidět rozdělení histogramu do 9 kanálů pro gradienty bez znaménka.

Tabulka 1: Rozdělení kanálů v histogramu

Kanály	1	2	3	4	5	6	7	8	9
Stupně	0	20	40	60	80	100	120	140	160

Do kanálů histogramu se přičte velikost gradientu v poměru podle jeho směru. Pokud by měl gradient směr 10° a velikost 4, tak se hodnota velikosti rozdělí napůl do kanálů 1 a 2, jelikož 10° leží přesně mezi 0° a 20°.

Velikosti gradientů se mění v rámci obrazu, jelikož jsou gradienty citlivé na změnu osvětlení. Proto se histogramy buněk spojí do větších histogramů bloků a tyto jsou následně normalizovány. Bloky se posouvají po obrazu velikosti buňky, takže se histogramy buněk budou ve výsledném histogramu opakovat. I když to zvětší paměťové nároky a počet operací, je to nezbytné pro optimální normalizaci. Normalizované histogramy se spojí dohromady a vznikne globální histogram, který popisuje obraz.

3.5 Klasifikátory

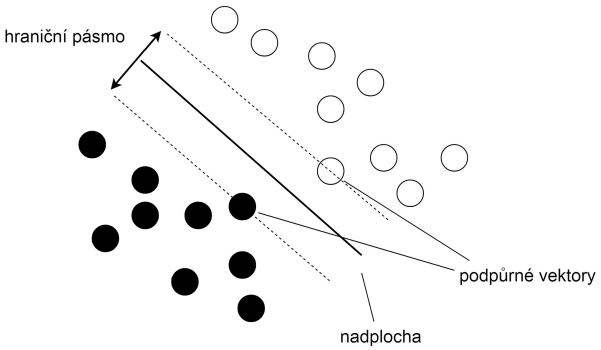
Po vytvoření nového prostoru příznaků trénovacích dat a projekci testovacího obrazu do tohoto prostoru výše zmíněnými metodami, lze obraz klasifikovat následujícími klasifikátory.

3.5.1 k-nejbližších sousedů

K-nejbližších sousedu (k-nearest neighbors) je jedna z nejjednodušších metod pro klasifikaci. Metoda vypočítá vzdálenost k nejbližších vektorů v prostoru příznaků od vektoru vstupního obrazu. Obraz je klasifikován za třídu, do které patří největší počet z k sousedů. Jako vzdálenost se často používá euklidovská metrika.
Metodu je možné rozšířit o nastavení prahu a použití vah. Prah určí do jaké vzdálenosti bude brát metoda v potaz vektory v prostoru a rozdělení vah vektorů se používá, aby bližší vektory měly vetší podíl na klasifikaci, než ty vzdálenější. Pokud $k=1$, tak se jedná o metodu nejbližšího souseda.

3.5.2 Metoda podpůrných vektorů

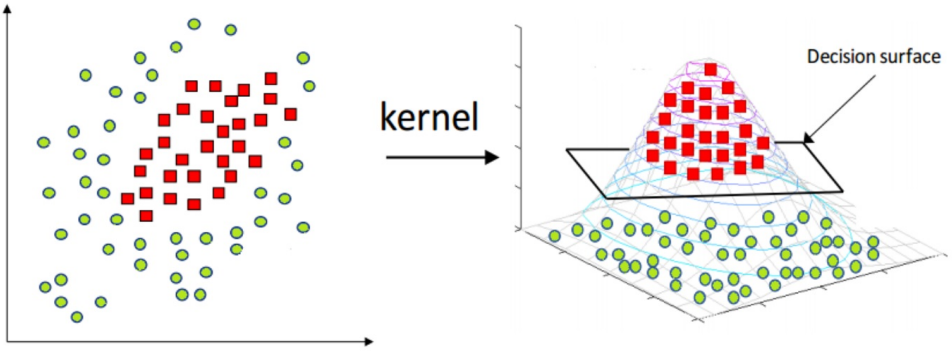
Základní verze metody podpůrných vektorů (Support vector machine) je binární lineární klasifikátor. Jméno metody je odvozeno z podpůrných vektorů, které jsou nejbližší vektory příznaků z jedné třídy ke druhé a určují pozici nadplochy. Úkolem metody je nalézt tuto nadplochu, resp. (k-1)-rozměrnou plochu v k-rozměrném prostoru, která rozdělí prostor příznaků trénovacích dat podle tříd. Pokud jsou data lineárně rozdělitelná, existuje nekonečno takovýchto nadploch. Ideální nadplocha by však měla mít co největší vzdálenost od podpůrných vektorů každé třídy a zároveň správně rozdělit všechny vektory. Vytvoří se tedy maximální hraniční pásmo ve kterém neleží žádné vektory a na okrajích leží podpůrné vektory (obrázek 12).



Obrázek 12: Hraniční pásmo v prostoru příznaků

Bohužel data nejsou v praxi tak jednoduše rozdělitelná. Proto se tolerují i špatně klasifikované vektory a snaží se najít co nejmenší cost funkce nadplochy, která bere v potaz velikosti hraničního pásma a počet špatně klasifikovaných vektorů. Nastavením hyperparametru C se určí penalizace za špatně klasifikované vektory.

Existují také data, která nejsou lineárně rozdělitelná. Z tohoto důvodu se SVM rozšiřují tzv. jádrovým trikem, s kterým se prostor příznaků transformuje do vyšší dimenze, kde jsou data lineárně rozdělitelná (obrázek 13).



Obrázek 13: Transformace prostoru příznaků do vyšší dimenze [19]

Jelikož se jedná o metodu binární, rozděluje pouze dvě třídy. Existují ale i přístupy, které používají kombinaci SVM, pro rozdělení třech a více tříd.

One-against-all

Aplikuje n SVM klasifikátorů u n tříd, kde existuje klasifikátor pro každou třídu, který rozdělí vektory příznaků dané třídy, proti vektorům všech ostatních tříd.

One-against-one

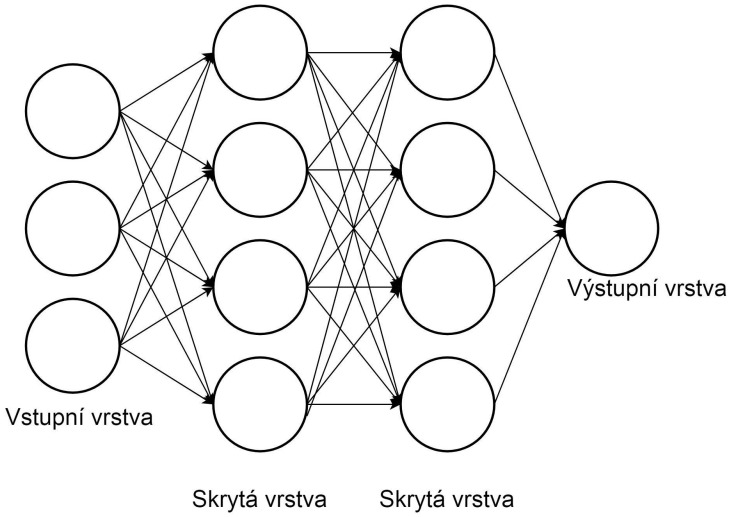
Aplikuje $\frac{n(n-1)}{2}$ SVM klasifikátoru u n tříd, kde pro každou kombinaci dvou tříd, existuje klasifikátor.

3.6 Umělé neuronové sítě

Umělé neuronové sítě (Artificial neural network) jsou odvětvím umělé inteligence, které se učí bez potřeby předešlé znalosti s jakými daty pracuje. V dnešní době jsou používané pro mnoho věcí, ale v této práci jsou důležité pro jejich schopnost klasifikace. Síť nejenom klasifikuje data, ale také provede všechno potřebné zpracování dat, jako extrakci příznaků a redukci dimenzí.

Síť je složená z neuronů, které jsou základní výpočetní jednotkou sítě. Neurony jsou rozdělené do vstupní, několika skrytých a výstupní vrstvy (obrázek 14), kde v každé vrstvě se nachází neurony se společnou charakteristikou. Mezi neurony jsou spojení, které vždy vedou do další vrstvy. Spojení mají váhy, které upravují posílanou hodnotu mezi neurony a informace se posílá ze vstupních neuronů přes spojení dále, až do výstupních neuronů. Posílané hodnoty v síti se vynásobí váhami spojení a pokud neutron přijímá více hodnot, tak se sečtou. Tyto váhy tedy rozhodují, které vstupní neurony jsou důležité pro výstupní neuron. Vypočtená hodnota se předá aktivační funkci.

Aktivační může být i jednoduchá step funkce, která se aktivuje a pošle dále hodnotu 1, pokud je hodnota vetší než určitý práh, nebo v opačném případě pošle hodnotu 0. Často používané funkce jsou nelineární, jelikož s lineární funkcí $A = cx$ i s nekonečno vrstvami, by byl výstup pouze lineární kombinací vstupů, takže by se síť chovala stejně, jako kdyby měla pouze jednu vrstvu. Často používaná nelineární sigmoid funkce $A = \frac{1}{1+e^{-x}}$, také oproti step funkci uhladí hodnotu do intervalu {0,1}. Neuronová síť samozřejmě neumí správně klasifikovat data v prvním cyklu, ale postupně se učí jako mozek. Učení se děje pomocí algoritmu zpětného šíření, který upravuje váhy sítě. Více informací o zpětném šíření v následující kapitole.



Obrázek 14: Ukázka struktury neuronové sítě

3.6.1 Konvoluční neuronové sítě

Konvoluční neuronové sítě (Convolutional neural network) jsou modelem hlubokých neuronových sítí, které se zejména používají u zpracování obrazů. Hluboké neuronové sítě mají mnoho skrytých vrstev. Tento model zakládá na faktu, že charakteristiky obličeje blíže k sobě, mají mezi sebou vetší souvislost než charakteristiky dále od sebe. Proto se používají hlavně u obrazů, jelikož při promíchání pixelů nedává obraz smysl. Všechny neurony mezi vrstvami tedy nejsou propojené. Výhoda tohoto faktu je menší výpočetní náročnost, ale také invariance umístění hledaného objektu.

V roce 1998 byla zveřejněná průkopnická architektura LeNet-5 [10], jedna z prvních CNN, která byla vyvíjena postupem několika opakování od roku 1988. Tato práce byla zaměřená na klasifikaci ručně psaných znaků a její architektura je základem pro mnoho CNN v dnešní době. Ale i po zveřejnění LeNet-5, CNN nebyly moc používány, až dokud nenastala "revoluce" v roce 2012 v podobě AlexNet [13]. AlexNet byl použit v soutěži zvané ImageNet Large Scale Visual Recognition Challenge (ILSVRC), pořádané každoročně od roku 2010. Soutěž slouží pro porovnání metod v detekci a rozpoznání objektů na podmnožině ImageNet databáze s 1000 kategoriemi. AlexNet obsadil v soutěži první příčku v klasifikaci s top-5 error rate 15,3%, s obrovským náskokem v porovnání s druhou příčkou s top-5 error rate 26,2% [11]. Za zmínku stojí, že AlexNet byl jediný CNN v soutěži. Pochopitelně po takovém velkém úspěchu se upoutala na CNN pozornost a již v následujícím roce byla většina práci založena na CNN. Na poslední soutěži v roce 2017, byl top-5 error rate snížen až na 2,25% [12].

Architektura CNN je složena z následujících vrstev, které se v architektuře opakují a jejich kombinace se liší podle navržení modelu.

Konvoluční vrstva je stavební kámen celého modelu a taky z něho vychází jeho název. Tato vrstva slouží pro extrakci příznaků. Každý neuron vrstvy má filtr resp. konvoluční jádro, které se přikládá na část jeho vstupu. Probíhá tedy skalární součin mezi vstupní maticí a matici filtru. Filtr se poté posouvá po celém vstupu a výsledkem je příznaková mapa. Počet příznakových map ve vrstvě je daný hloubkou vrstvy. Každá vrstva sdílí společnou charakteristiku, která se stává více komplexnější a méně lokálně prostorná hlouběji v síti. První vrstvy mohou detekovat hrany a další nízko úrovňové příznaky, zatímco hlubší vrstvy detekují tvary jako například oči a nos. Hyperparametr teto vrstvy je již zmíněná hloubka, která určí počet aplikovaných filtrů, dále krok určí po kolika pixelech se bude filtr posouvat a zero-padding doplní vstupní matici na krajích nulami, aby se mohl filtr aplikovat i u pixelů na okraji.

Jako aktivační funkce, která zajistí nelineárnost sítě, se v dnešní době často používá **Rectified Linear Unit (ReLU)** operace $f(x) = max(0, x)$. Funkce nahradí všechny negativní hodnoty příznakové mapy nulou a používají se u výstupu konvolučních a plně propojených vrstev. Síť s ReLU funkcemi se v porovnání s sigmoid, nebo tanh funkcemi trénuje rychleji [13]. Navíc

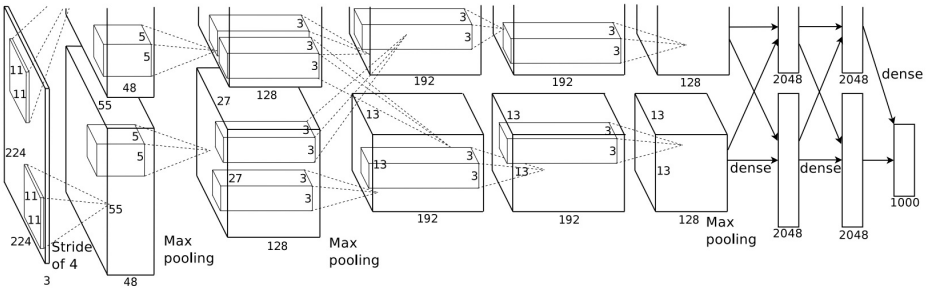
také předchází problému zvaný *vanishing gradient problem.*

Pooling vrstva, typicky aplikovaná po konvoluční vrstvě, slouží pro redukci prostorové velikosti zmenšením příznakové mapy a také invarianci otočení a posunutí objektu. Používají se max-pooling, nebo také average-pooling funkce. Lepších výsledků ale dosahuje max-pooling, který rozdělí vstupní obraz do oblastí a z hodnot v této oblasti vybere pouze tu největší. Hyperparametry této vrstvy nastaví krok, velikost a tvar oblasti.

Na konci sítě jsou **plně propojené vrstvy**. Ty se dají představit jako regulérní neuronové sítě, napojené po konvolučních a pooling vrstvách. Tudíž každý neuron je napojený na každý neuron předchozí vrstvy a celá vrstva by se dala reprezentovat jako vektor. Vrstvy vytvoří nelineární kombinaci z příznaků, které vytvořili konvoluční vrstvy.

Poslední plně propojená vrstva je **výstupní vrstva**, která používá jako aktivační funkci Softmax funkci, místo funkce ReLU. Výsledek funkce je N dimenzionální vektor, kde N je počet tříd. Každá hodnota vektoru reprezentuje pravděpodobnost dané třídy hodnotou mezi 0–1. Součet všech hodnot činí 1.

Na obrázku 15 je vidět architektura AlexNet, složená z 5 konvolučních vrstev a 3 plně propojených vrstev. Na výstupu každé vrstvy je aplikována ReLU funkce. Max-pooling vrstva používá 3x3 oblasti s krokem 2 pixelů. Výstup poslední plně propojené vrstvy je rozdělený pomocí Softmax funkce do 1000 kategorii [13].



Obrázek 15: Struktura archiektury AlexNet, kde je síť rozdělená do dvou částí, kvůli trénování na dvou GPU [13]

Zpětné šíření

Filtry konvolučních vrstev ze začátku neví jaké tvary mají hledat. Jsou tedy nastaveny náhodně a poté upraveny pomocí zpětného šíření. Nejprve trénovací obraz prošel síti a ve výstupní vrstvě vznikl vektor pravděpodobnosti rozdělení tříd. Teď se s vektorem a skutečnou třídou, do které

obraz patří, vypočítá cost funkce. Často je používána Mean squared error funkce:

$$E = \frac{1}{n}\sum_{i=1}^{n}(t_i - y_i)^2$$

kde t je výstup sítě a y je požadovaný výsledek.
Cost funkce vyjadřuje míru chyby sítě. Celková chyba sítě se zpětně šíří v síti a vypočítává se chyba u každé váhy. Následně se vypočítá *gradient descent* chybové funkce, za cílem nalezení lokálního minima. Váhy jsou podle výsledného gradientu upravené, ve snaze minimalizace cost funkce. Problém může nastat, pokud je lokální minimum o hodně "výše" než globální minimum. Další problém zvaný *vanishing gradient problem*, již zmíněn u ReLU funkce, se stává u hlubších sítí, kde zpětně šířený gradient se postupem do hlubších vrstev zmenšuje a váhy poslední vrstvy mohou zůstat beze změny.

4 Použitý hardware

Použité zařízení s i.MX procesorem na kterém proběhlo veškeré testování je Raspberry Pi 3 model B a jako kamera byl vybrán kamerový modul Raspberry Pi v2.

4.1 i.MX

i.MX je rodina mikrokontrolérů od firmy NXP, zaměřující se na nízkou spotřebu. Jejich široké uplatnění zahrnuje chytré telefony, jednodeskové počítače, a dotykové panely do aut a výtahu. Tyto mikrokontroléry jsou založené na ARM architektuře. Výroba série i.MX 1 se odstartovala v roce 2001, založená na ARM920T architektuře s kmitočtem 100–200 MHZ. Novější série, specificky i.MX série 5 a výše, používají jádra ARM Cortex série, která se rozdělují na ARM Cortex-A, ARM Cortex-M a ARM Cortex-R. ARM Cortex-A je skupina 32 a 64-bitových procesorových jáder, určené pro použití u aplikací ve výkonných systémech. ARM Cortex-M je skupina 32-bitových jáder, určená pro použití v mikrokontrolérech a širokou škálu zabudovaných systému. ARM Cortex-R jsou jádra používaná u aplikací v reálném čase, ale u i.MX procesorů nejsou použity. Série i.MX 8M byla ohlášena 4.ledna 2017. Je zaměřena pro aplikace zpracovávající hlas, zvuk a video [22]. Dostupné verze této série jsou jedno, dvou a čtyř jádrové ARM Cortex-A53 s kmitočtem 1,5GHZ. Všechny verze také obsahují jedno jádro Cortex-M4F pro podporu zpracování v reálném čase.

4.2 Raspberry Pi 3 model B

Raspberry Pi je série jednodeskových počítačů vyvinuté pro podporu vyučování informatiky ve školách a v rozvojových zemích. Je ale hojně používán pro domácí automatizaci, díky jeho nízké ceně. Jako první vyšel Raspberry Pi 1 model B v únoru 2012. Pro tuto práci je použitý Raspberry Pi 3 model B, který vyšel v únoru 2016 a pohybuje se v ceně 1000Kč. Je 10x výkonnější než Raspberry Pi 1 [20] a používá již zmíněnou 4 jádrovou verzi i.MX 8M série, která je vhodná právě pro zpracování videa. Počítač je vybaven HDMI, 3.5mm jack, RJ-45, CSI a 4x USB porty a oproti starším verzím také WiFi a Bluetooth rozhraním [21]. Jako úložiště dat slouží microSD. Doporučený operační systém je Raspbian, který je založený na Debianu a byl přímo vyvinut pro Raspberry počítače. Jako desktopové prostředí je použita modifikovaná verze LXDE, nazývaná PIXEL. Pro tuto práci je použita verze Raspbian Stretch.

4.3 Kamerový modul Raspberry Pi v2

Raspberry Pi obsahuje CSI konektor pro použití Raspberry Pi kamerového modulu v2. Kamera podporuje natáčení videa v kvalitě 1080p30, 720p60 nebo 640x480p90 a pořízení fotografii o rozlišení až 8MPx. Pro použití kamery jsou dostupné oficiální knihovny MMAL, V4L2 (Video4Linux 2) a v této práci je použita právě knihovna V4L2.

5 Implementace

Implementace proběhla v jazyce C++ s použitím OpenCV knihovny.

5.1 OpenCV knihovna

Open source Computer Vision je multiplatformní knihovna s otevřeným zdrojovým kódem. Zaměřuje se zejména na počítačové vidění a zpracování obrazu v reálnem čase. V roce 1999 byla zveřejněna první alfa verze firmou Intel a od té doby proběhlo mnoho vylepšení a stále vychází pravidelně nové verze. Nejnovější verze je 3.4.1 a pro tuto práci je použita verze 3.3.0. Knihovna je napsána a vyvíjena v C++ s podporou pro C/C++, Python, MATLAB a Java rozhraní. Je široce používána v podnicích, výzkumných skupinách, vládních organizacích či široké veřejnosti s celkově více jak 17 milióny stáhnutími [23].

5.1.1 Použité moduly

OpenCV má strukturu modulů, kde každý modul má jinou funkcionalitu.
imgproc (image processing) obsahuje funkce pro zpracování obrazu.

- lineární a nelineární filtrace obrazu
- geometrické transformace
- ekvalizace histogramů
- konverze barevného prostoru

highgui (high-level GUI) poskytuje rozhraní pro vizualizaci výsledku atd.

- vytvoření a manipulaci oken
- čtení a zápis obrazu z paměti či disku
- čtení videa z kamery či souboru a zápis videa do souboru

ml (machine learning) je modul zabývající se strojovým učením a obsahuje třídy a metody pro klasifikaci, regresní analýzu a shlukování.

- Metoda podpůrných vektorů
- Algoritmus k-nejbližších sousedů
- Vícevrstvý perceptron (neuronové sítě)

objdetect (object detection) je modul pro detekci objektů.

- Kaskáda klasifikátorů
- HoG deskriptor

Existují také "extra" moduly, které nejsou součástí oficiální OpenCV distribuce. Většinou obsahují nové moduly, které zatím nejsou dobře vyzkoušené, nebo nemají stabilní API. Pro rozpoznání obličejů je potřeba jeden z těchto modulů, zvaný **face** (face analysis).

5.2 Viola-Jones

U implementace metody je použitý modul *objdetect*, pro vytvoření třídy *CascadeClassifier*. Do této třídy načteme kaskádu klasifikátorů s funkcí *load* (výpis 1). OpenCV obsahuje již předtrénované kaskády klasifikátorů v XML souborech pro detekci obličeje, očí, těla a podobně. Pro detekci obličeje použijeme *haarcascade_frontalface_alt.xml*. Následně načteme testovací obraz jako matici do kontejneru cv::Mat. Načtený obraz převedeme do stupňů šedi s funkcí *cvtColor* a pro úpravu kontrastu provedeme ekvalizaci histogramů s funkcí *equalizeHist* (výpis 2).

```
string face_cascade_name= " haarcascade_frontalface_alt.xml ";
CascadeClassifier face_cascade;
if(!face_cascade.load(face_cascade_name)) { cout << "Chyba pri nacitani kaskad.
    " << endl;
return -1 };
```

Výpis 1: Načtení kaskád klasifikátorů

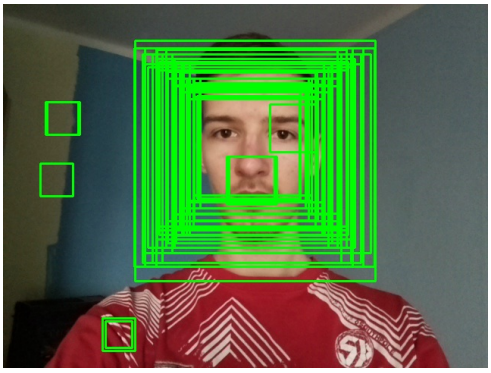
```
cvtColor(inputImage, grayImage, CV_BGR2GRAY);
equalizeHist(grayImage, grayImage2);
```

Výpis 2: Předzpracování obrazu

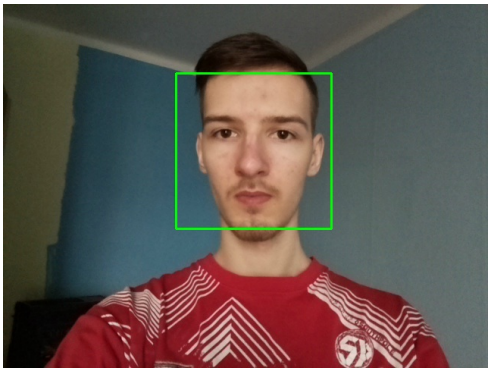
Pro samotnou detekci obličejů použijeme funkci

void cv::CascadeClassifier::detectMultiScale(cv::Mat image, vector<cv::Rect> &faces, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size(), Size maxSize=Size())

kde *image* bude *grayImage2* z výpisu 2. Pokud budou detekovány obličeje, funkce vrátí pro každý obličej jeho pozici jako 4 body obdelníku ve výstupním parametru *faces*. Důležité parametry této funkce, které ovlivní průběh detekce, jsou *scaleFactor* a *minNeighbors*.
Hodnota *scaleFactor* nastaví o kolik se zmenší velikost obrazu poté, co detekční okno dokončí svůj cyklus. Například při *scaleFactor=1.1* se obraz zmenší pokaždé o 10%.
Hodnota *minNeighbors* nastaví kolik objektů musí být detekovaných v blízkosti detekovaného objektu, aby byl prohlášen za obličej. Při detekci s *minNeighbors=0* dojde k velkému počtu falešných detekcí (obrázek 16a). Na obrázku 16b je vidět, že stačí pouze *minNeighors=2* a počet falešných detekcí prudce klesne.



(a) minNeighbors=0



(b) minNeighbors=2

Obrázek 16: Porovnání detekce obličeje při změně parametru minNeigbors

5.3 FaceRecognizer

K implementaci rozpoznání obličeje se v OpenCV používá abstraktní třída *FaceRecognizer* z modulu *face*. Třída *FaceRecognizer* vychází z base třídy *Algorithm*, která je používána pro všechny algoritmy v OpenCV. V našem případě poskytuje jednotný přístup algoritmům Eigenfaces, Fisherfaces a LBPH. Všechny algoritmy používají funkci *create* pro vytvoření modelu, *train* pro natrénování modelu a *predict* pro klasifikaci. U trénování i predikce se očekávají obrazy ve stupních šedi. Pro klasifikaci je použita metoda nejbližšího souseda (1-NN) s euklidovskou metrikou.

5.3.1 Eigenfaces

Pro vytvoření Eigenface modelu použijeme funkci

static Ptr<EigenFaceRecognizer> cv::face::EigenFaceRecognizer::create(int num_components=80, double threshold=DBL_MAX)

kde *num_components* je počet hlavních komponent s největší vlastní hodnotou ponechaných z algoritmu PCA. Maximální počet komponent je počet trénovacích obrazů.
Po klasifikaci s metodou 1-NN je výsledkem vzdálenost a *threshold* nastaví hranici, po kterou se testovací obraz bude považovat za obličej. Z toho vyplývá, že čím menší *threshold* resp. vzdálenost, tím je klasifikace striktnější.

Následně se daný model natrénuje na trénovacích datech funkcí

void cv::face::FaceRecognizer::train(vector<cv::Mat> images, vector<cv::int> labels)

kde *images* obsahují množinu testovacích obrazů a *labels* jsou množina označení obrazů.

Na natrénovaném modelu můžeme klasifikovat vstupní obraz *src* s funkcí

void cv::face::FaceRecognizer::predict(cv::Mat src, int &label, double &confidence)

kde *confidence* je sebejistota algoritmu, že se jedná o daný obličej. Tato hodnota je euklidovská vzdálenost stejně jako *threshold*, takže čím menší je hodnota, tím větší je sebejistota algoritmu. Pokud *confidence>threshold*, tak *label* vrátí hodnotu -1. Pokud *confidence<threshold*, tak *label* vrátí označení daného obličeje. Všechny trénovací a testovací obrazy musí mít stejnou horizontální i vertikální velikost.

5.3.2 Fisherfaces

Pro vytvoření Fisherface modelu použijeme funkci

static Ptr<FisherFaceRecognizer> cv::face::FisherFaceRecognizer::create(int num_components=0, double threshold=DBL_MAX)

kde jediný rozdíl oproti funkci *create* u Eigenfaces je parametr *num_components*. U Fisherfaces jsou *num_components* komponenty z LDA a maximální počet komponent je počet tříd - 1. Nejlepší je nechat si všechny komponenty, nastavením hodnoty *num_components=0*. Všechny trénovací i testovací obrazy musí mít stejnou horizontální i vertikální velikost.

Funkce *train* a *predict* jsou zcela stejné jako u metody Eigenfaces.

5.3.3 LBPH

Pro vytvoření LBPH modelu použijeme funkci

static Ptr<LBPHFaceRecognizer> cv::face::LBPHFaceRecognizer::create(int radius=1, int neighbors=8, int grid_x=8, int grid_y=8, double treshold=DBL_MAX)

Tento LBPH model používá kruhovou LBP verzi, ve které *radius* značí vzdálenost sousedů od prostředního pixelu a *neighbors* stanovuje počet sousedů v daném okolí. Hodnoty *grid_x* a *grid_y* značí velikost bloků.

U LBPHFaceRecognizer jde použít funkce *update*, pomocí které se dá aktualizovat již natrénovaný model, tzn. dají se přidat nové obrazy. Tohle u třídy *EigenFaceRecognizer*, ani *FisherFaceRecognizer* není možné.

Funkce *train* a *predict* jsou zcela stejné jako u metody Eigenfaces.

6 Experimenty

Pro objektivní zhodnocení a nalezení optimálních parametrů, je přesnost a rychlost u detekce a rozpoznání odděleně otestována na statických databázích. Poté s použitím těchto parametrů a kamera modulem, je zkombinovaná detekce a rozpoznání obličejů v reálném prostředí.

6.1 Úspěšnost detekce obličejů

Pro otestování metody Viola-Jones, byla použitá databáze Face detection data set and benchmark [14]. Celá databáze obsahuje 2845 obrazů s 5171 obličeji. Pro naše testování bylo použito 67 obrazů s 100 obličeji. Obrazy jsou barevné, s obličeji zachycenými v reálných podmínkách (obrázek 17). Tyto podmínky zahrnují různé pózy, malé rozlišení, nezaostření a překrytí obličejů. Rozlišení obrazů se liší, ale přibližně se pohybuje kolem 350x450 px.

Obrázek 17: Ukázky z FDDB databáze

U detekce mohou nastat 4 možné výsledky:

- **true positive** (TP) - správně klasifikován pozitivní vzorek, resp. správně detekovaný obličej
- **false positive** (FP) - nesprávně klasifikován negativní vzorek, resp. nesprávně detekovaný obličej
- **true negative** (TN) - správně klasifikován negativní vzorek, resp. správně zahozené detekční okno
- **false negative** (FN) - nesprávně klasifikován pozitivní vzorek, resp. nesprávně zahozené detekční okno

Přesnost detekce je měřena s **F1-score**, jenž se používá pro měření u binárních klasifikátorů a nepoužívá pro vyhodnocení true negatives. To je v našem případě výborné, jelikož je u detekce obličejů obrovský počet true negatives. Měření bere v úvahu *precision* (precizi) a *recall* (úplnost).
Precision hodnotí detekci v závislosti na počtu obličejů z množiny detekovaných objektů. Takže *precision* klesá s narůstajícím počtem false positives.

TP/TP+FP
Recall hodnotí detekci v závislosti na počtu detekcí z množiny obličejů. Takže *recall* klesá s narůstajícím počtem false negatives.
TP/TP+FN
F1-score se vypočítá harmonickým průměrem *precision* a *recall*.

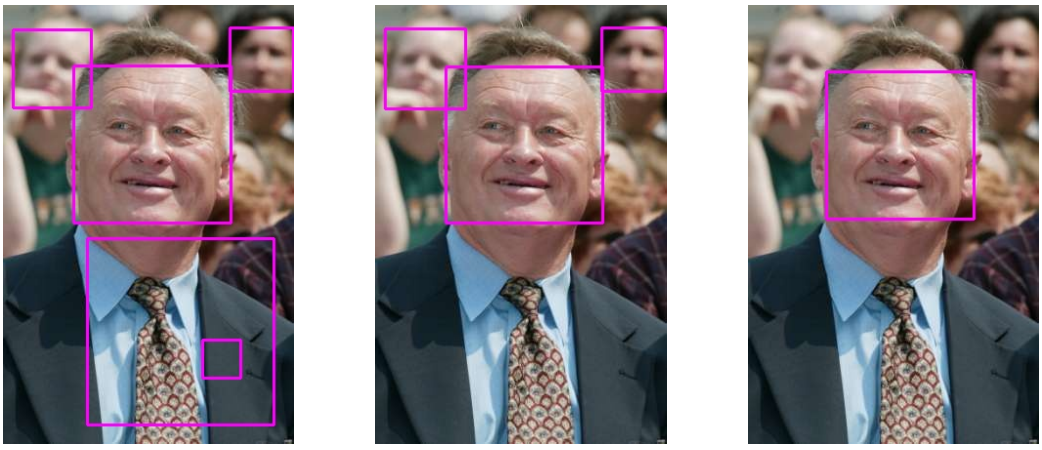
$$F1 = \frac{(2 * precision * recall)}{(precision + recall)}$$

Čas v tabulce 2 značí průměrnou dobu detekcí obličejů v jednom obraze.

Tabulka 2: Viola-Jones detekce s minNeighbors=2

scaleFactor	Precision	Recall	F1-score	Čas [s]
1,01	0,4943	0,88	0,633	3,09
1,05	0,8673	0,85	0,8585	0,6547
1,1	0,9318	0,82	0,8723	0,3337
1,2	0,9743	0,76	0,8539	0,2089
1,3	0,96	0,72	0,8228	0,1461
1,4	0,7582	0,69	0,7225	0,0866

V tabulce 2 můžeme sledovat, že při zmenšování *scaleFactor* se konstantně zlepšuje *recall*, ale také roste doba detekce a po určitém bodě se začne zhoršovat *precision*. Tato změna je také zobrazena na obrázku 18. Nejlepší F1-score je zaznamenám při *scaleFactor=1,1* a *minNeighbors=2*, kde je optimální balance mezi počtem false positives a negatives.
Změna *minNeighbors* nebyla zobrazena, jelikož nebyly zaznamenány výrazné změny. Důležité je však vždy aplikovat *minNeighbors>0*.



(a) scaleFactor=1.01 (b) scaleFactor=1.1 (c) scaleFactor=1.4

Obrázek 18: Porovnání detekce obličeje při změně parametru scaleFactor

6.2 Úspěšnost rozpoznání obličejů

U testování metod pro rozpoznání obličeje je použitá AT&T databáze, která obsahuje 40 subjektů s 10 vzorky pro každého, v rozlišení 92x112 px [24]. Obličeje jsou zachycené pod různým osvětlením a s různými výrazy, ve vzpřímeném čelním postoji s mírnou rotací hlavy (obrázek 19).

Obrázek 19: Ukázky z AT&T databáze

Pro ohodnocení modelu se databáze musí rozdělit na trénovací a testovací data. Model se natrénuje na trénovacích datech a poté bude použit pro klasifikování testovacích dat. Pro přesnější ohodnocení modelu a zkontrolování jestli náš model není přetrénovaný, je použita **k-násobná křížová validace**. Validace tedy lépe ohodnotí jak by se vedlo modelu v praxi. Metoda náhodně rozdělí data do k částí. Jedna část bude reprezentovat testovací data a *k-1* částí bude reprezentovat trénovací data. Tento proces bude opakován k-krát, aby se všechna data jednou staly testovacími. Pokud k=n, kde n je počet vzorků v databázi, tak se jedná o *leave-one-out křížovou validaci*. Konečné ohodnocení se vypočítá provedením aritmetického průměru na k výsledcích. Pro testování je použita 10-násobná křížová validace (obrázek 20).

Obrázek 20: Znazornění 10-násobné křížové validace [25]

Čas trénování v následujících tabulkách znamená průměrnou dobu trénování jednoho modelu a čas klasifikace znamená průměrnou dobu rozpoznání jednoho obličeje. Na čas strávený nad

klasifikací by se měl dávat větší ohled než na čas strávený nad trénováním, jelikož je stěžejní při rozpoznávaní v reálném čase. Trénování se na druhou stranu může provést na začátku pouze jednou, nebo může být také model předtrénovaný, například v XML souboru.
Jak je vidět v tabulce 3, po 50 komponentech u metody Eigenface již nenarůstá úspěšnost a pouze narůstá doba trénování a klasifikace.

Tabulka 3: Eigenfaces

num_components	Úspešnost [%]	Čas [s]	
		Trénování	Klasifikace
5	84	57,69	0,0015
10	94,25	60,96	0,0029
20	96,25	62,68	0,0082
50	97,75	65,67	0,0182
80	97,5	68,58	0,0278
200	97	80,99	0,0661

V tabulce 4 je vidět, že u metody Fisherfaces při maximálním počtu komponent bylo dosaženo nejlepší úspěšnosti, přičemž doba trénování příliš nenarůstá a doba klasifikace je více než přijatelná.

Tabulka 4: Fisherfaces

num_components	Úspešnost [%]	Čas [s]	
		Trénování	Klasifikace
10	84,75	79,62	0,0030
20	90,25	81,13	0,0082
30	92	83,38	0,0092
0	94,5	85,52	0,0147

U LBPH metody je v tabulce 5 a 6 vidět, že při zvětšování počtu sousedů, výrazně narůstá doba klasifikace. Dokonce při počtu sousedů více jak 8, model již nemusí být vhodný pro rozpoznání v reálném čase. Doba trénování také narůstá, ale v porovnání s Eigenfaces a Fisherfaces je o hodně kratší. Při zmenšení bloku na 4x4 se zmenší čas klasifikace a také se zlepší úspěšnost. Nejlepší výsledek dosahuje LBPH s počtem sousedů 8 a blokem 4x4. Hodnota radius byla nastavena na 1 a jeho změna nebyla zobrazena, jelikož nebyly zaznamenány výrazné změny.

Tabulka 5: LBPH s grid_x=8 a grid_y=8

num_components	Úspešnost [%]	Čas [s]	
		Trénování	Klasifikace
2	94	1,681	0,0135
4	98,25	2,536	0,0411
8	97,75	4,294	0,3938
12	97,75	9,638	5,2867

Tabulka 6: LBPH s grid_x=4 a grid_y=4

num_components	Úspešnost [%]	Čas [s]	
		Trénování	Klasifikace
2	93,25	1,279	0,0064
4	99	2,002	0,0141
8	99,25	3,689	0,1208
12	99	5,982	1,3257

6.3 Detekce a rozpoznání obličejů v reálném prostředí s kamerovým modulem Raspberry Pi v2

Pro rozpoznání obličejů s kamerovým modulem je AT&T databáze rozšířena o 3 subjekty se stejným počtem vzorků a podobným osvětlením a výrazy jako ve zbytku databáze. Pro oříznutí obličejů u nově přidaných subjektů, byla provedena detekce obličeje stejně jako u testování. Toto oříznutí proběhlo i u originálních subjektů AT&T databáze a v databázi zůstali pouze subjekty, u kterých byly detekovány všechny obličeje. Konečná databáze obsahuje 30 subjektů.

Testování proběhlo na nově přidaných subjektech v interních podmínkách. Poprvé za denního světla a poté večer při vnitřním osvětlení. Každý subjekt byl natáčen po dobu 90 snímků, při kterých byl rotován čelem ke kameře o 90° na obě strany od zdroje osvětlení. U každého videa je na každém snímku nejprve provedena detekce metodou Viola-Jones a pokud je detekován obličej, tak je provedeno rozpoznání obličeje. Toto testování je navrženo pro otestování invariance vůči osvětlení a jeho směru.

Rozlišení kamery bylo nastaveno na 640x480 px a jako parametry metod byly použity ty nejúspěšnější z testování v předešlých kapitolách.

- **Viola-Jones** - scaleFactor=1.1, minNeigbors=2
- **Eigenfaces** - num_components=50
- **Fisherfaces** - num_components=0
- **LBPH** - radius=1, neigbors=8, grid_x=4, grid_y=4

Hodnota *threshold* nebyla u rozpoznání nastavena.

V tabulkách 7 a 8 jsou výsledky zobrazeny jako průměrné hodnoty na všech testovaných subjektech u jednotlivých metod. Detekce za večerních podmínek měla 13 false positives a žádné false negatives, takže bylo provedeno rozpoznání u každého obličeje. Rozpoznání na false positives se nepočítaly do výsledků. Výsledky v tabulce 7 jsou velice dobré a u Eigenfaces a LBPH jsou skoro srovnatelné s výsledky v kapitole 6.2. U metody Fisherfaces jsou výsledky horší, ale pořád jsou relativně přijatelné.

Tabulka 7: Rozpoznání obličejů za večerních podmínek

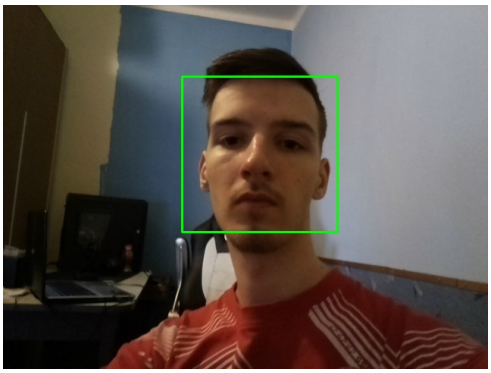
Metoda	Úspěšnost [%]	Čas klasifikace [s]
Eigenfaces	90	0,0182
Fisherfaces	81,11	0,0084
LBPH	92,96	0,097

Detekce za denních podmínek měla 12 false positives a 1 false negatives, takže bylo rozpoznání provedeno na 269 z 270 obličejů. Rozpoznání na false positives se nepočítaly do výsledků. V tabulce 8 jsou vidět o hodně horší výsledky vzhledem k výsledkům v tabulce 7, obzvlášťe u metody Eigenfaces. Tohle zhoršení je pravděpodobně způsobeno změnou osvětlení oproti trénovacím obrazům, které byly nafoceny ve večerních podmínkách.

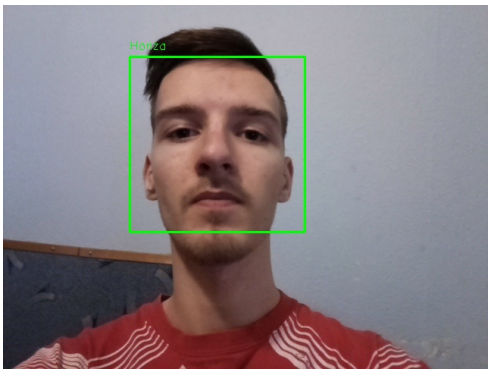
Tabulka 8: Rozpoznání obličejů za denních podmínek

Metoda	Úspěšnost [%]	Čas klasifikace [s]
Eigenfaces	27,88	0,0179
Fisherfaces	45,72	0,009
LBPH	75,46	0,095

Obličeje s čelním postojem ke zdroji osvětlení byly obvykle správně klasifikovány, ale při postoji kde byla většina obličeje překryta stínem, byly výsledky horší (obrázek 21 a 22).

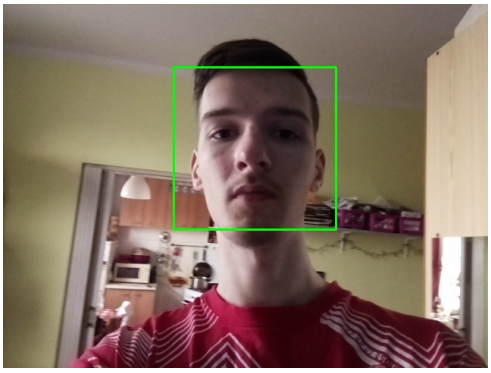


(a) Nesprávně klasifikovaný obličej

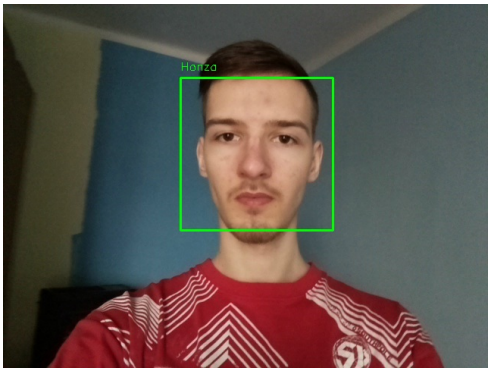


(b) Správně klasifikovaný obličej

Obrázek 21: Porovnání rozpoznání obličeje ve večeních podmínkách při změně směru osvětlení



(a) Nesprávně klasifikovaný obličej



(b) Správně klasifikovaný obličej

Obrázek 22: Porovnání rozpoznání obličeje za denních podmínkek při změně směru osvětlení

Celkový čas procesu strávený na rozpoznání obličeje je čas detekce + čas klasifikace + čas ostatních funkcí (tabulka 9). Klasifikace i detekce zabrala přibližně stejnou dobu u obou podmínek, kde čas detekce je průměrně 0,5s, čas ostatních funkcí je 0,04s a čas klasifikace je vidět v tabulkách 7 a 8. Celkový čas tedy hlavně záleží na času detekce, který můžeme snížit zmenšením hodnoty *scaleFactor* na úkor přesnosti.

Tabulka 9: Počet zpracovaných snímků za sekundu, při procesu rozpoznání obličeje v rámci jednotlivých metod

Metoda	Počet snímků za sekundu
Eigenfaces	1,79
Fisherfaces	1,82
LBPH	1,57

7 Závěr

Úkolem práce bylo nastudovat a popsat metody pro detekci a rozpoznání obličejů. Následně implementovat metody s pomocí knihovny OpenCV na zařízení s procesorem i.MX a ověřit jejich rychlost a přesnost.

Implementované metody byly Viola-Jones, Eigenfaces, Fisherfaces a LBPH v jazyce C++ na zařízení Raspberry Pi 3 model B. U metody Viola-Jones byl použit soubor *haarcascades_frontal_alt.xml* z knihovny OpenCV, ve kterém jsou natrénované kaskády klasifikátorů. Následně byla metoda otestována na podmnožině databáze FDDB, kde dosáhla F1-score 0,8723, což je slušný výsledek vzhledem k obtížnosti dané databáze. Metody Eigenfaces, Fisherfaces a LBPH byly natrénovány i otestovány na databázi AT&T pomocí 10-násobné křížové validace. Všechny metody dosáhly velice dobrých výsledků s úspěšností nad 94,5%. Nejúspěšnější nastavení metod bylo použito při testování s kamerovým modulem Raspberry Pi v2. Trénování proběhlo na AT&T databázi s třemi nově přidanými subjekty, na kterých bylo provedeno testování. Tyto testy byly jednodušší pro detekci než u databáze FDDB, avšak hlavním účelem bylo naměřit úspěšnost klasifikace u rozpoznání obličejů. Nejhoršího výsledku dosáhla metoda Eigenfaces za denních podmínek s úspěšností 27,88% a nejlepší výsledek měla metoda LBPH za večerních podmínek s úspěšností 92,96%.

Možné rozšíření této práce by bylo použití neuronových sítí nebo vytvoření vlastní databáze s více vzorky pro každý subjekt.

Literatura

[1] Ming-Hsuan Yang, D. J. Kriegman, and N. Ahuja. *Detecting faces in images: a survey.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 1, pp. 34–58, January 2002.

[2] P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR'01), vol. 1, pp. 511-518, 2001.

[3] Paul Viola and Michael J. Jones. *Robust real-time face detection.* International Journal of Computer Vision, vol. 57, no. 2, pp. 137–154, May 2004.

[4] M. A. Turk and A. P. Pentland. *Face recognition using eigenfaces.* 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR'91), pp. 586–591, June 1991.

[5] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. *Eigenfaces vs. fisherfaces: recognition using class specific linear projection.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 711–720, July 1997.

[6] T. Ahonen, A. Hadid, and M. Pietikainen. *Face description with local binary patterns: Application to face recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 12, pp. 2037–2041, December 2006.

[7] Matti Pietikäinen, Abdenour Hadid, Guoying Zhao, and Timo Ahonen. *Local Binary Patterns for Still Images.* Computer Vision Using Local Binary Patterns, pp. 13–47, January 2011.

[8] Rahim, M. A. et al. *Face Recognition using Local Binary Patterns (LBP).* Global Journal of Computer Science and Technology. vol. 13, 2013. ISSN 0975-4350. Dostupné z: https://globaljournals.org/GJCST_Volume13/1-Face-Recognition-using-Local.pdf.

[9] N. Dalal and B. Triggs. *Histograms of oriented gradients for human detection.* 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, pp. 886–893, June 2005.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, November 1998.

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge.* International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015.

[12] Jie Hu, Li Shen, and Gang Sun. *Squeeze-and-excitation networks*. CoRR, abs/1709.01507, 2017.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. *Imagenet classification with deep convolutional neural networks*. 25th International Conference on Neural Information Processing Systems (NIPS'12) - Volume 1, pp. 1097–1105, USA, 2012. Curran Associates Inc.

[14] Vidit Jain and Erik Learned-Miller. *FDDB: A benchmark for face detection in unconstrained settings*. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[15] *RAPP: Integral Images* [online]. 1. June 2016 [cit. 2018-04-15]. Dostupné z: http://www.nongnu.org/rapp/doc/rapp/integral.html

[16] POWELL, Victor a Lewis LEHE. *Principal Component Analysis explained visually* [online]. 12. February 2015 [cit. 2018-04-15]. Dostupné z: http://setosa.io/ev/principal-component-analysis/

[17] GUO, Bob. *PCA (4) : LDA (Linear Discriminant Analysis)* [online]. 21. January 2017 [cit. 2018-04-15]. Dostupné z: https://algorithmsdatascience.quora.com/PCA-4-LDA-Linear-Discriminant-Analysis

[18] PIETIKÄINEN, Matti. *Local Binary Patterns* [online]. Scholarpedia, 5(3):9775, 2010 [cit. 2018-04-15]. Dostupné z: http://www.scholarpedia.org/article/Local_Binary_Patterns

[19] RASHMI, Jain. *Simple Tutorial on SVM and Parameter Tuning in Python and R* [online]. HackerEarth, 21. February 2017 [cit. 2018-04-15]. Dostupné z: https://www.hackerearth.com/blog/machine-learning/simple-tutorial-svm-parameter-tuning-python-r/

[20] UPTON, Eben. *Raspberry Pi 3 on sale now at $35* [online]. 29. February 2016 [cit. 2018-04-15]. Dostupné z: https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/

[21] *Raspberry Pi model B* [online]. [cit. 2018-04-15]. Dostupné z: http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf

[22] *Press Release: NXP i.MX 8M Processor Poised to Transform IoT Audio, Voice and Video Interactions* [online]. NXP Semiconductors, 4. January 2017 [cit. 2018-04-15]. Dostupné z: http://media.nxp.com/phoenix.zhtml?c=254228&p=irol-newsArticle&ID=2233904

[23] *OpenCV Download statistics* [online]. Slashdot Media, c2018 [cit. 2018-04-15]. Dostupné z: https://sourceforge.net/projects/opencvlibrary/files/stats/timeline?dates=2001-09-20+to+2018-04-15

[24] *The Database of Faces* [online]. AT&T Laboratories Cambridge, c2002 [cit. 2018-04-15]. Dostupné z: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[25] ROSAEN, Karl. *Pipeline gotchas, k-fold cross-validation, hyperparameter tuning and improving my score on Kaggle's Forest Cover Type Competition* [online]. 20. June 2016 [cit. 2018-04-15]. Dostupné z: http://karlrosaen.com/ml/learning-log/2016-06-20/

A Obsah přiloženého CD

Elektronická verze této dokumentace ve formátu LaTeX a PDF
Zdrojové kódy aplikace
Makefile soubor pro zkompilování kódu a vytvoření spustitelné aplikace
Trénovací a testovací data
Postup pro zkompilování a nainstalování OpenCV knihovny na nově nainstalovaném operačním systému Raspbian